UzhYarpModules v0.2

In Preparation of the Software Workshop on

March 2, 2007

February 12, 2007

Jonas Ruesch

University of Zurich
Department of Informatics
Artificial Intelligence Laboratory

Contents

1 Introduction **2**
1.1 Yarp Module Definition 2
1.2 Module Configuration 2
1.3 Graphical User Interfaces 2
1.4 Yarp Data Exchange Formats 3
1.5 Portability 3

2 Major Module OTTO **3**

3 Minor Modules **3**
3.1 Object Detection Based on Motion 3
3.2 Object Detection Based on Color 4
3.3 Eyes and Head Control 4

4 Development Team **4**

1 Introduction

The in the following described modules are currently under development and are targeted at solving the task "Object Tracking Through Occlusion" (OTTO). Ideally the solution to this task will consist of several generally applicable modules.

1.1 Yarp Module Definition

If we talk about Yarp modules, we are thinking of standalone executables which read input data from a Yarp network port and write output data back to another Yarp network port. If this definition is in accordance with other development teams has to be verified.
At the moment we are working with ports handling images (reading/writing) and motor control (reading encoder values/writing motor commands). For exchanging arbitrary data between Yarp modules there will be the need to define customized data formats. The decision at what level functionality should be packaged as a Yarp module is in my point of view quite essential and I would be interested how other development teams decide on this.

1.2 Module Configuration

Every Yarp module can be configured at startup and at runtime using a Yarp Property object. At startup this Property object is created by reading from a configuration file located at a default location. (e.g. /.uzhICub/-modXY.ini). If a different configuration file than the one at default location is required the filepath and/or filename can be specified using command-line arguments. At runtime the applied Property object can be created by a graphical user interface.

1.3 Graphical User Interfaces

Finally our goal is to provide for each module a graphical user interface (GUI) for runtime configuration purposes. The user can decide wether he wants to use the GUI or not setting the according flag in the configuration file. To keep maximum flexibility, the decision to use the GUI can be taken at compile time too. This removes all GUI-library dependencies for anybody not wanting to use graphical user interfaces. This capability is already available and accessible via the cmake configuration dialog.
Data entered in the GUI is stored in a Yarp Property object and is applied to the module in the same way as any other module configuration described above.

Graphical user interfaces are developed using QT from Trolltech which is a multiplatform toolkit available in open source versions for Windows and Linux.

1.4 Yarp Data Exchange Formats

A key point in inter-module communication is to agree on the exchanged data format (e.g. content of exchanged Yarp "bottles"). I think in our case it will be necessary to decide for example on a Yarp bottle format representing visually tracked objects. Such data might then be passed between clustering modules, trajectory tracking modules and any motor/eye control module.

1.5 Portability

Currently development is done under Linux. However, we are aiming at providing Linux and Windows compatible source code for all of the final modules.

2 Major Module OTTO

The main "Object Tracking Through Occlusion" module: "The ability to track objects through occlusion by extrapolation of motion, only saccading after the expected transition time behind the occluder".

Input: Images grabbed from a Yarp port.

Output: A suggested gaze point in images coordinates written to the Yarp network as the content of a Yarp bottle.

3 Minor Modules

Modules which contribute to the major module OTTO but can be used independently.

3.1 Object Detection Based on Motion

Clustering of motion fields using a given motion threshold.

Input: Images grabbed from a Yarp port.

Output: List of detected objects along with acquired data (e.g. average moving direction, absolute motion value of the detected motion field).

3.2 Object Detection Based on Color

Clustering of color fields using a given hue and the desired hue tolerance.

Input: Images grabbed from a Yarp port.

Output: List of detected objects along with acquired data (e.g. average moving direction, absolute motion value of the detected motion field).

3.3 Eyes and Head Control

We aim at developing an eye/head control module. The main goal is the ability to saccade to a given location in image coordinates and to do smooth pursuit of a target location. There is certainly a big overlap between head control code (Genova, Lissbon, Lausanne) and we would appreciate to unify the already available software.

Input: Target location in image coordinates (x,y).

Output: Reading/writing from/to a Yarp (remote) controlboard.

4 Development Team

Currently involved with software development for RobotCub at the Artificial Intelligence Laboratory in Zurich are:

Jonas Ruesch, PhD student, ruesch at ifi.unizh.ch

Yvonne Gustain, diploma student, yvonne_g at gmx.net